Istituto Italiano di Tecnologia

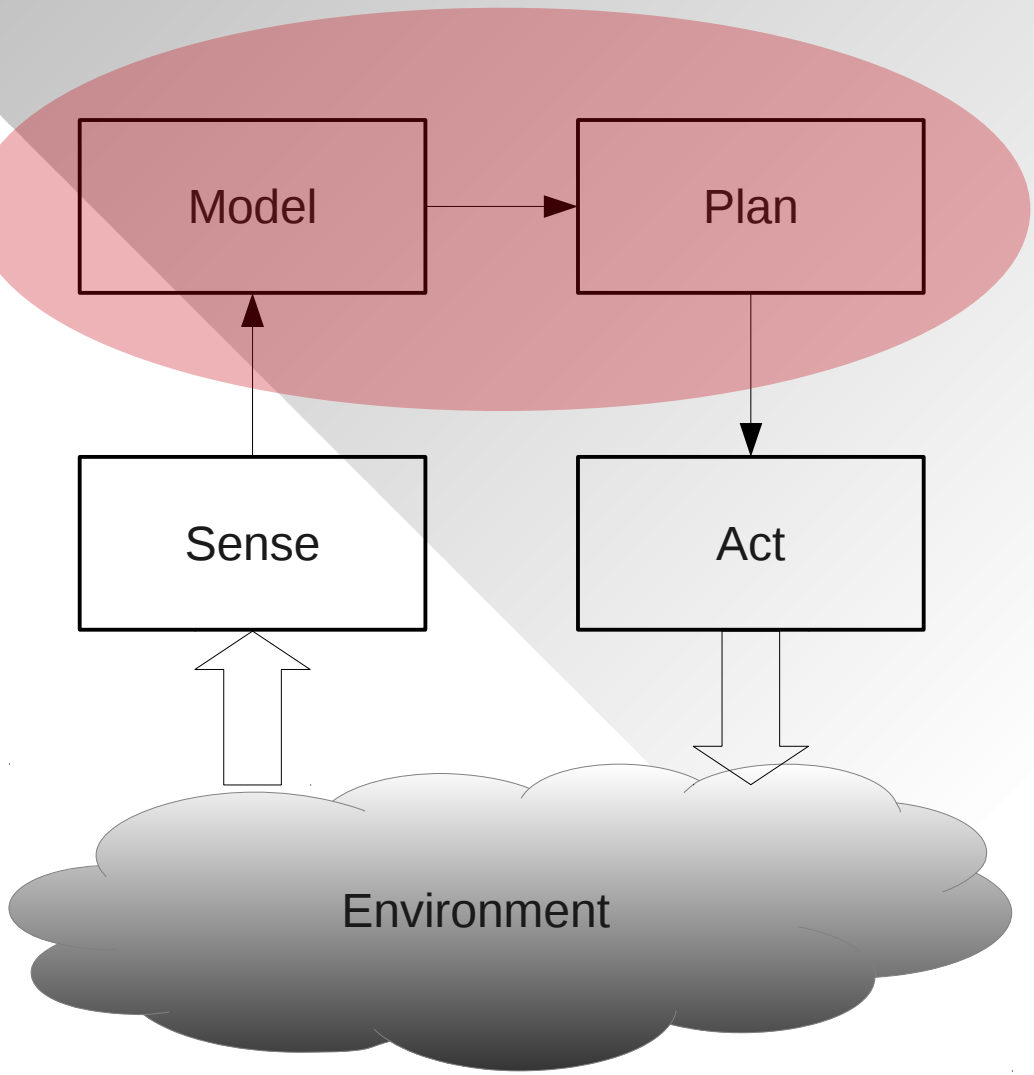
A port–arbitrated mechanism for distributed behavior selection in robotics

Ali Paikan (iCub Facility - Italian Institute of Technology)

VVV13, Veni Vidi Vici 2013, the iCub Summer School

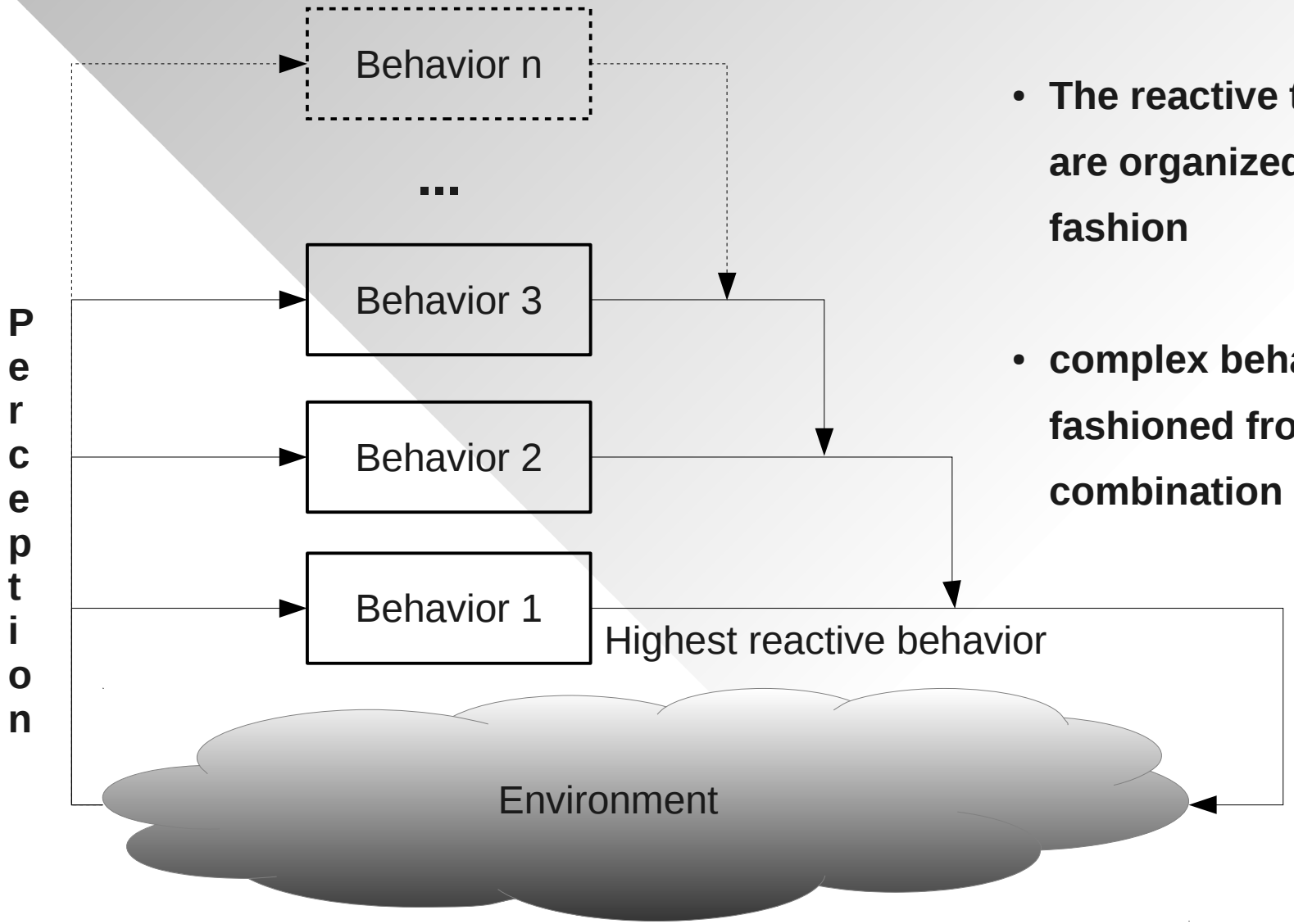
July 22th 2013 – Sestri Levante

Sense-Plan-Act



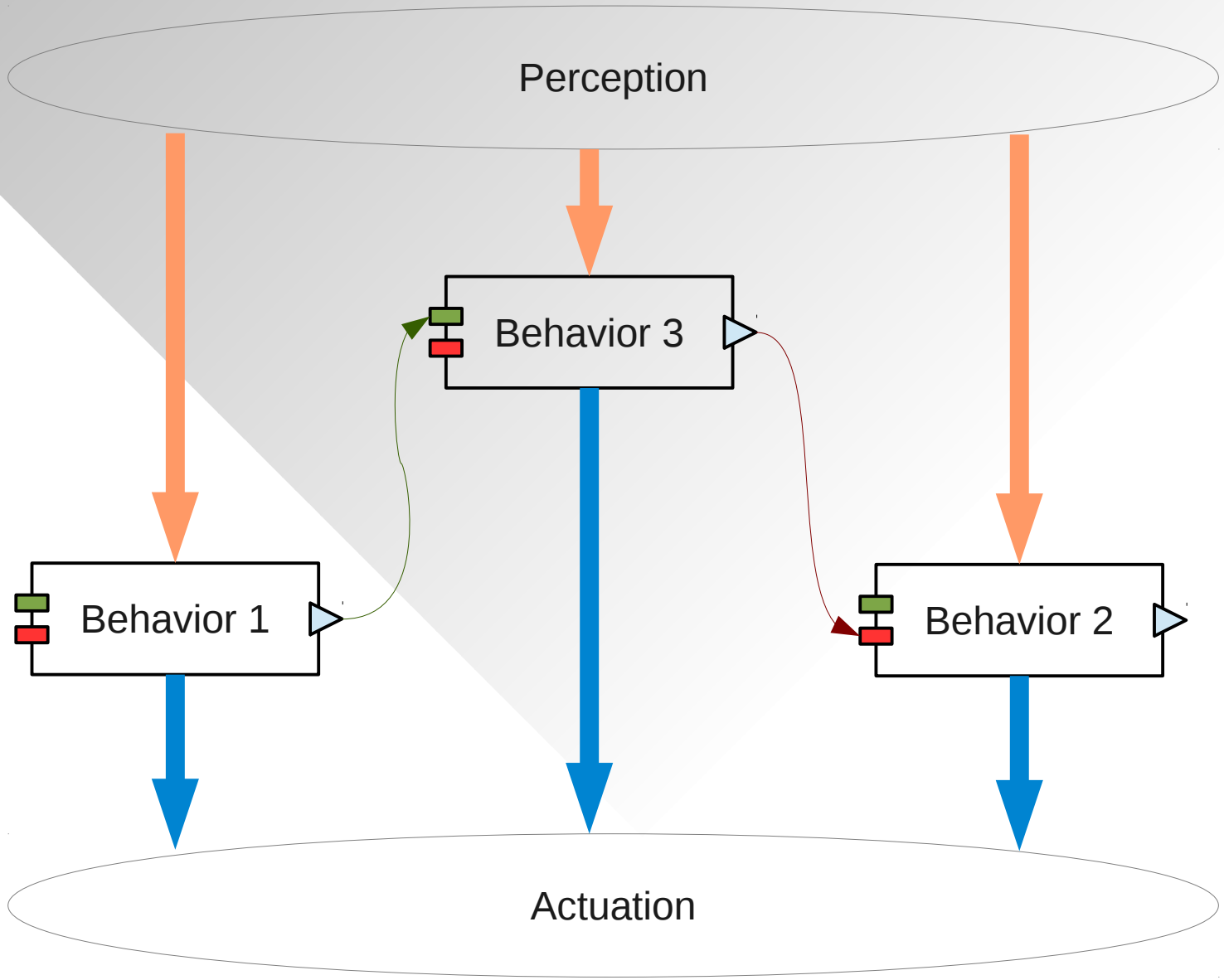
- **Inherently Sequential**
- **Environment is changing during modeling/planing**
- **Reactiveness depends on how fast modeling/planing is performed**
- **...**

Brook's Subsumption Architecture

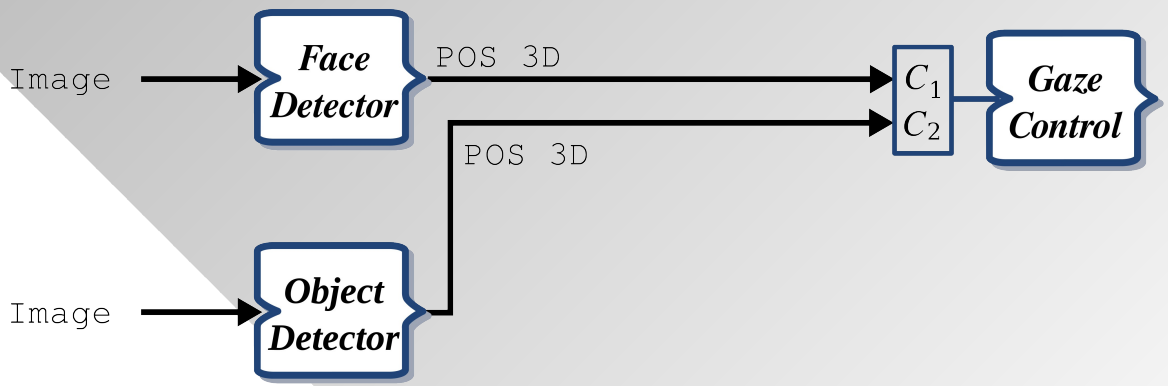


- **Inherently parallel system**
- **The reactive tasks or behaviors are organized in a bottom-up fashion**
- **complex behaviors are fashioned from the combination of simpler one**

Behavior-based system



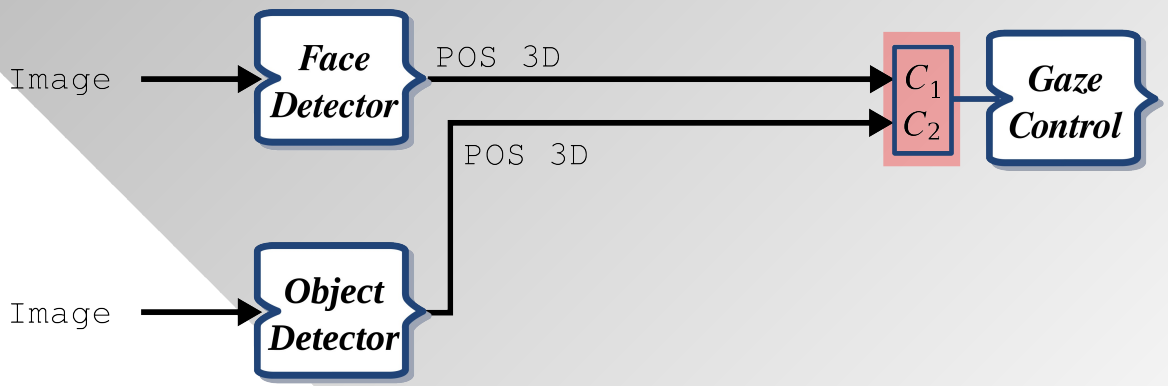
Coordination using port arbitration



- *'Face Detector'* and '*Object Detector*' can both send 3D position data to '*Gaze Control'* which controls the robot's head to gaze accordingly.
- Since there is no synchronization among modules, data can be delivered to the input port of '*Gaze Control*' at any time, potentially causing conflicts.

A coordination mechanism should be employed to avoid conflict between these competitive connections!

Coordination using port arbitration

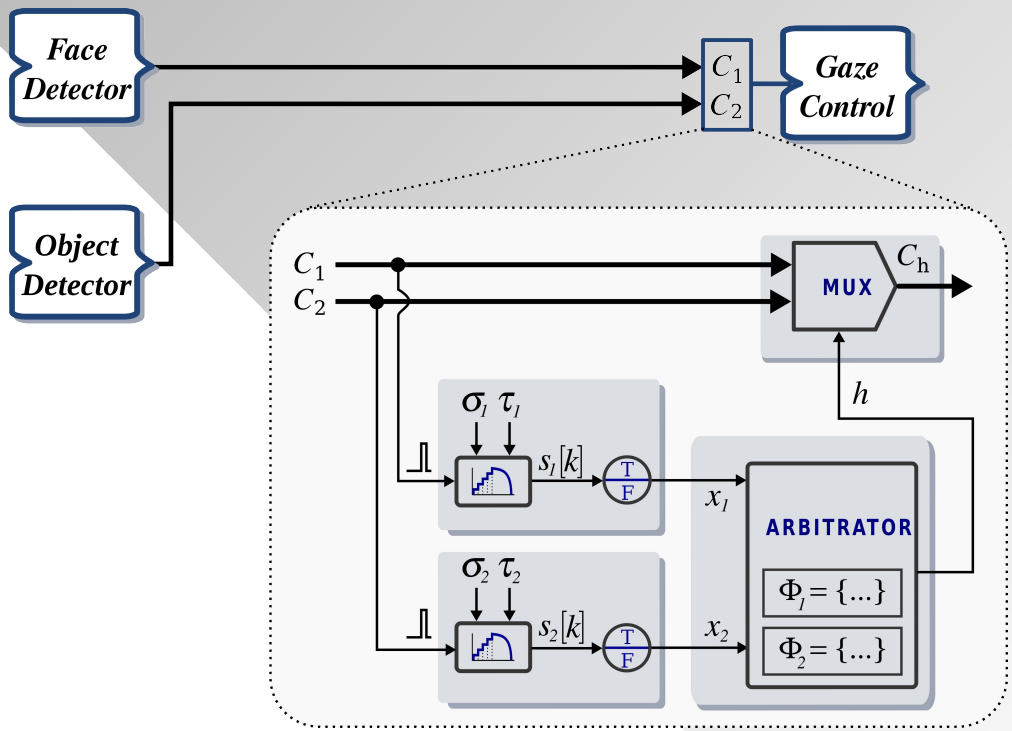


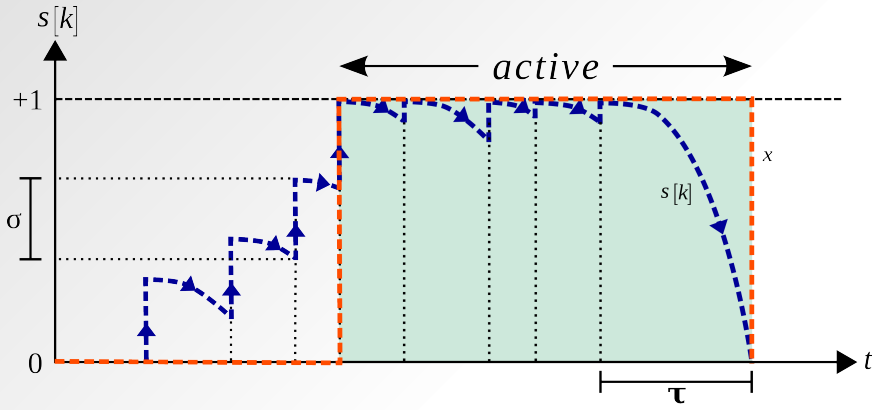
- If '*Gaze Control*' receives data from C1, robot will track a face.
- If '*Gaze Control*' receives data from C2, robot will track an object.

Imagine we want the robot to track the face if there is no object in the scene:

`"SELECT connection C1 IF C2 does NOT send any data"`

Coordination using port arbitration





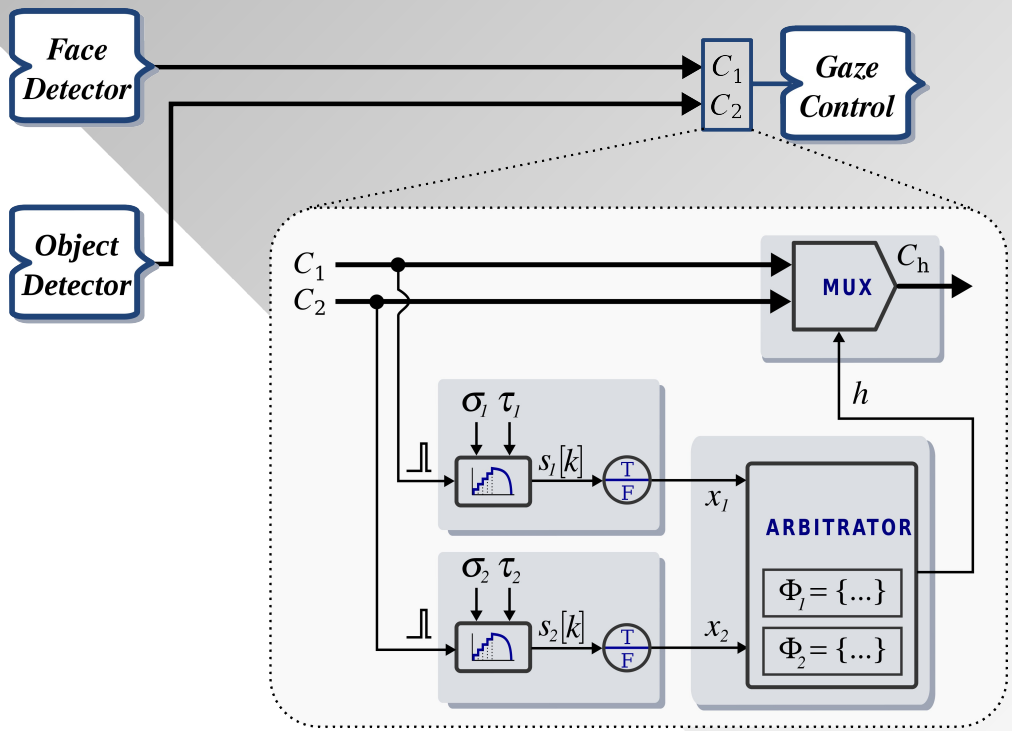
Track the face if there is no object in the scene:

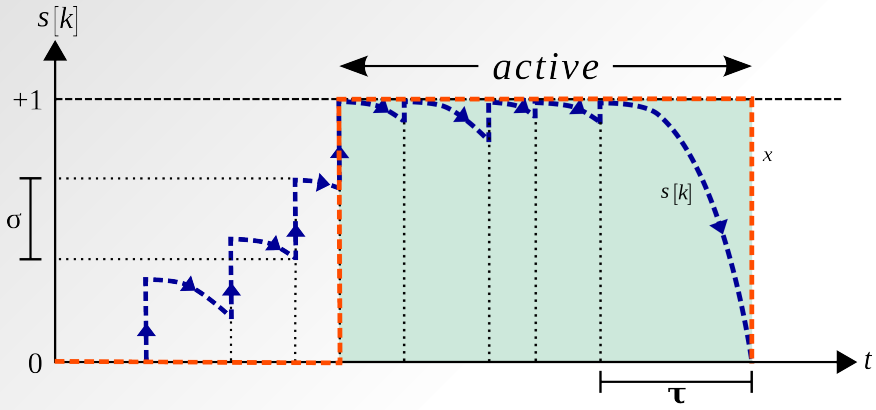
Φ1 : C1 AND NOT C2

Φ2 : C2

Inhibition
Excitation

Coordination using port arbitration





Track the object if there is ALSO a person in the scene:

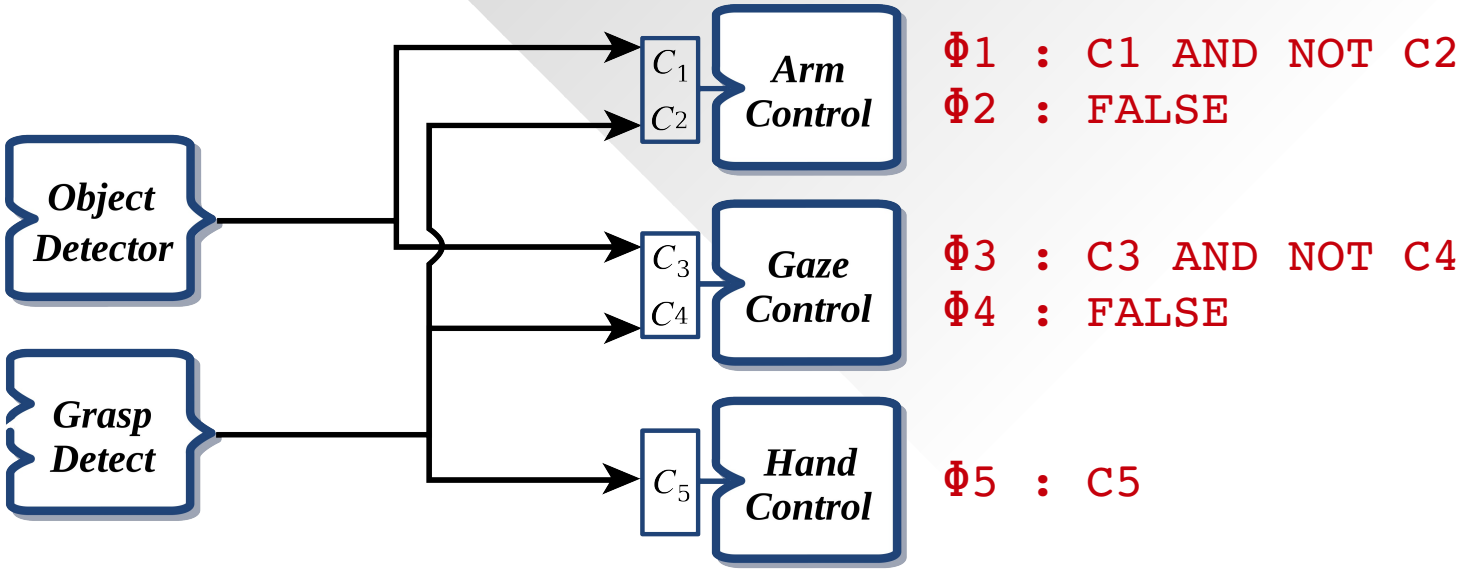
Φ1 : FALSE

Φ2 : C2 AND C1

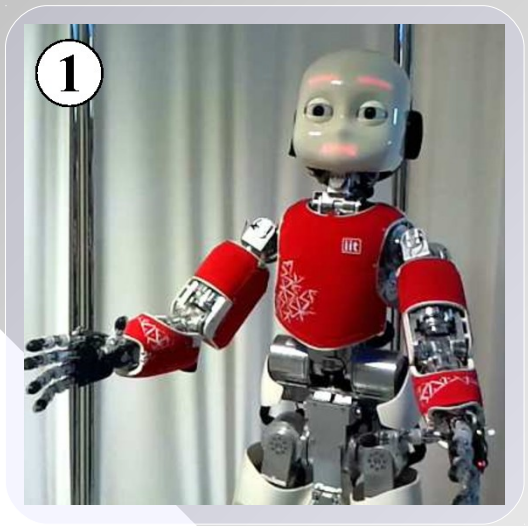
More complex example (Catch)

Track an object and grasp it:

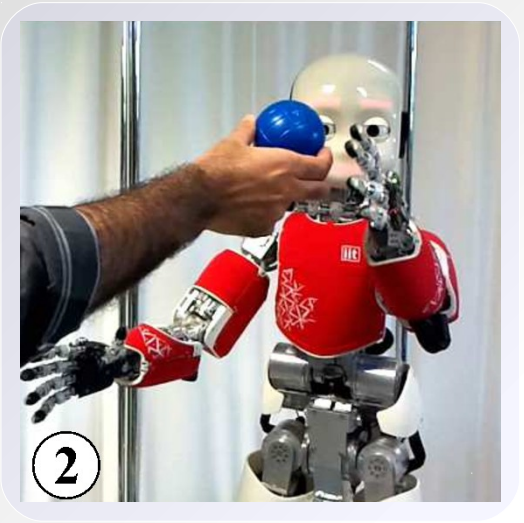
- Try to reach for object by hand and follow it by head
- Continuously check if the object is close enough for grasp
- When grasping the object, inhibit the movement of arm and head of the robot



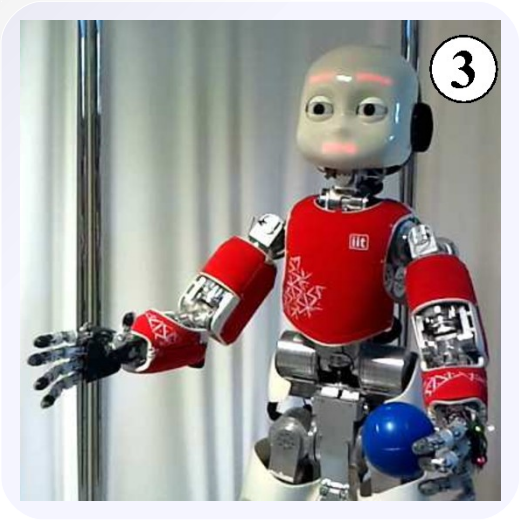
More complex example (Catch)



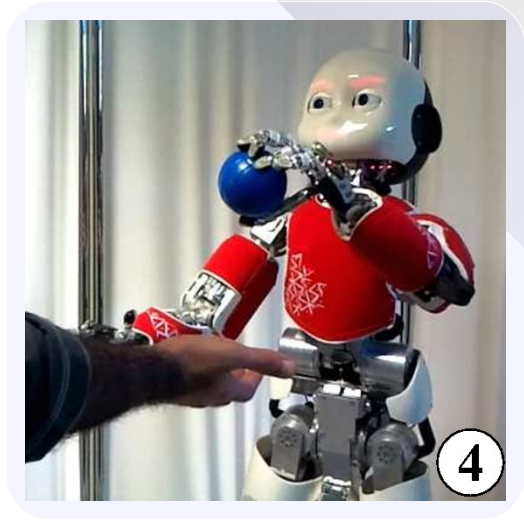
Look for object



Take

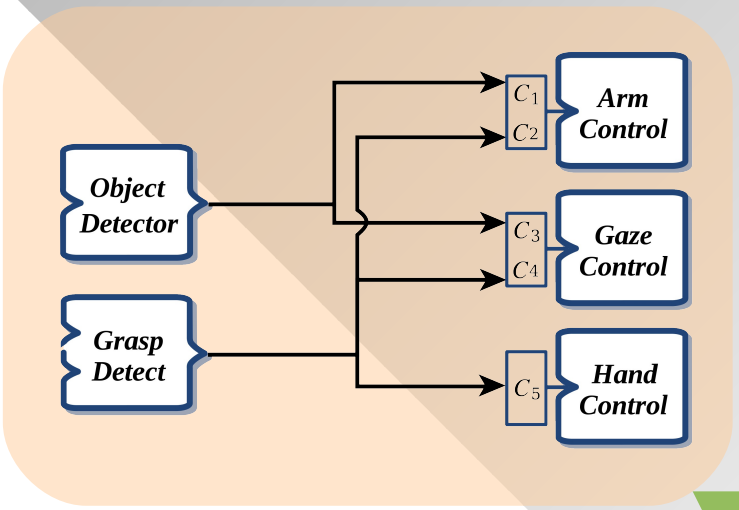


Look for face

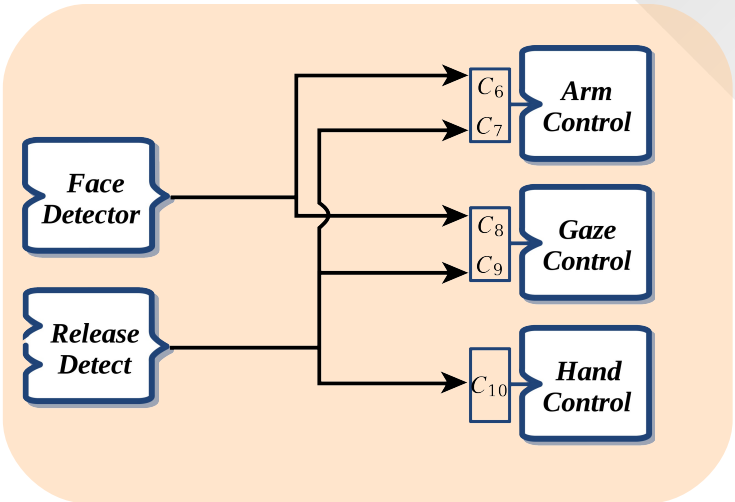


Return

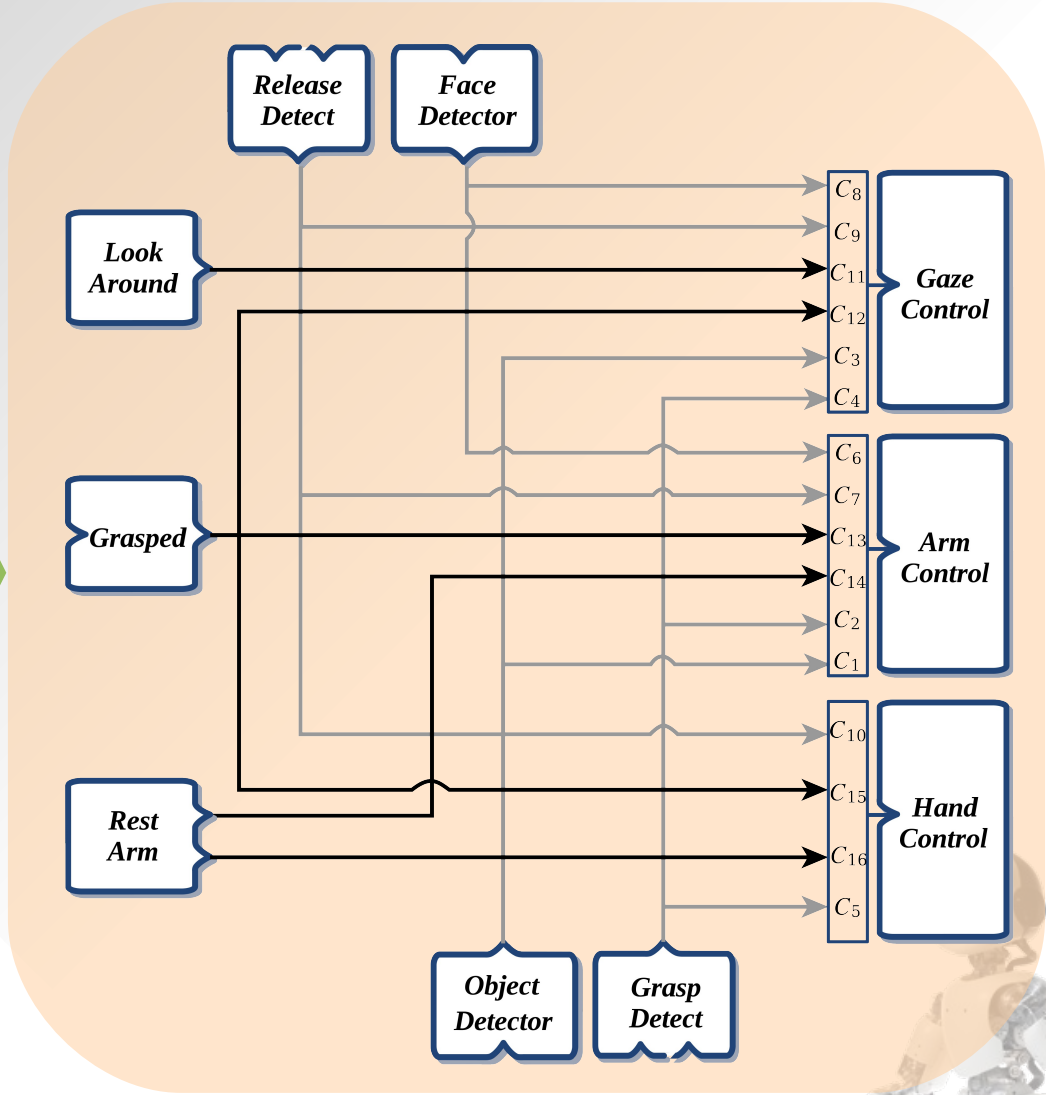
Catch and Return scenario



Catch



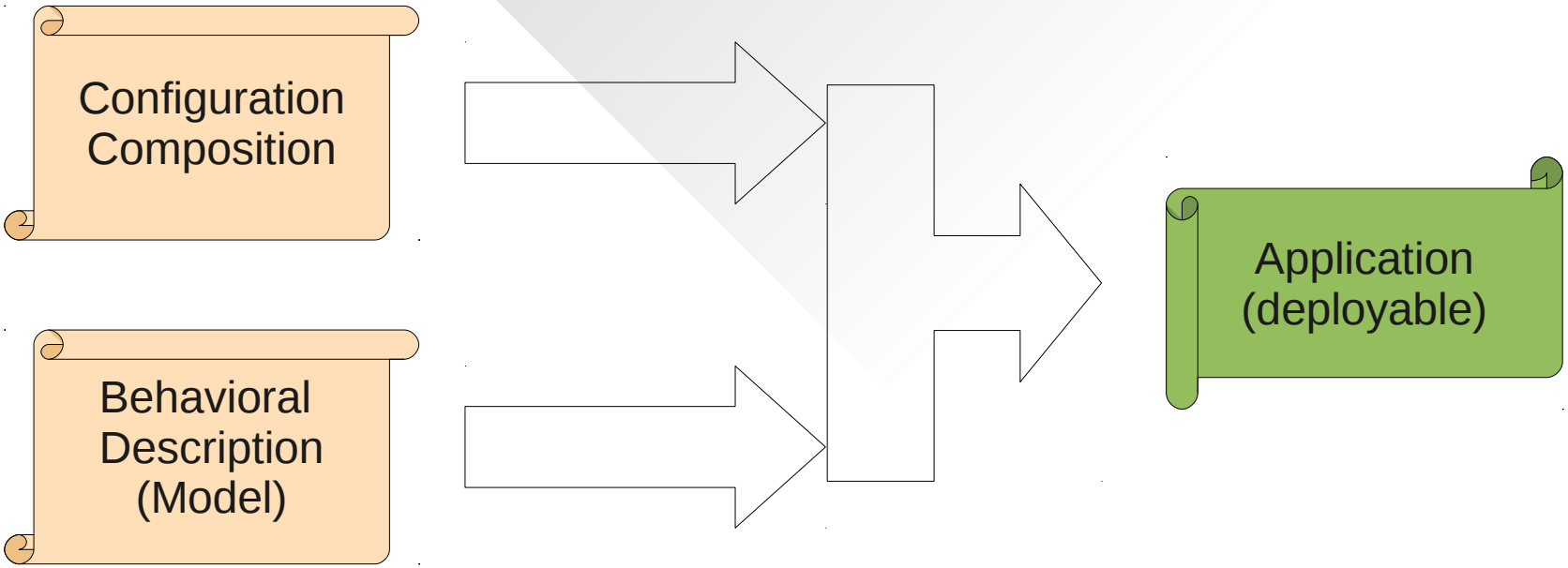
Return



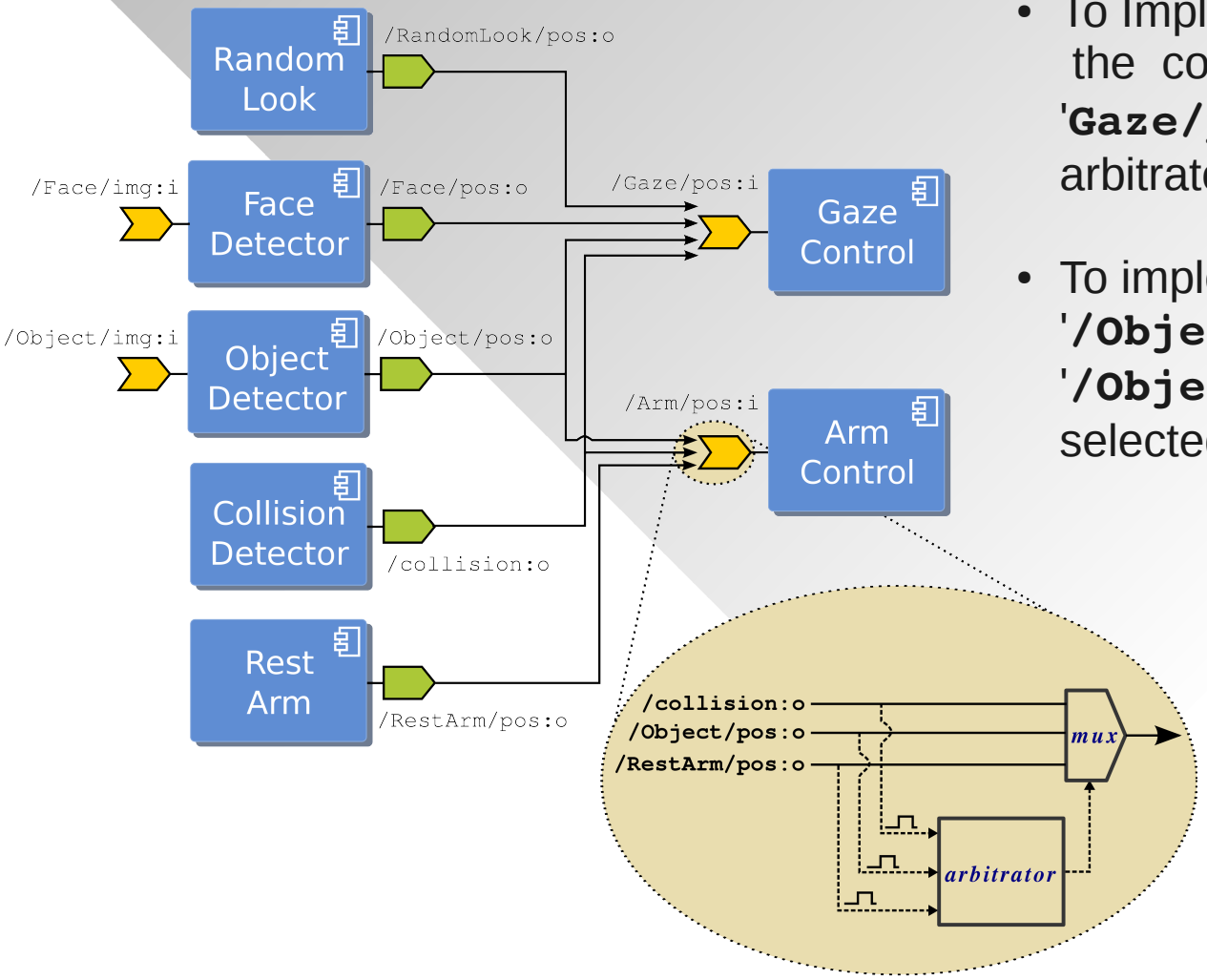
Catch and Return

Modeling the behaviors

- Best practices in robotics, promotes the idea that composition and coordination of software component should be separated during the software development life cycle.
- We need to separate representation of the behaviors from the composition of the software components.
- Based on different behavioral descriptions, the same software components can be reused to implement different applications.



Modeling the behaviors



- To Implement a behavior called '*Follow Face*', the connection from '**`/Face/pos:o`**' to '**`Gaze/pos:i`**' should be selected by port arbitrator.
- To implement 'Track Object' behavior, '**`/Object/pos:o`** to **`/Gaze/pos:i`**' and '**`/Object/pos:o`** to **`/Arm/pos:i`**' should be selected by port arbitrator.

Modeling the behaviors

- ***Configuration*** of a behavior is the list of connections which should be selected by the port arbitrators to implement the behavior.
- ***Condition*** is an optional property which specifies in, first-order logic, a constraint that should be verified for the behavior to be activated.
- ***Inhibition***, specifies inhibitions between behaviors. Specifying inhibitions allows coordinating behaviors that are competing for the same resources.
- Behaviors can be grouped to describe A ***Meta behavior.***

Track Object

Condition: ¬ `/collision:o`

Configuration:

`/Object/pos:o` -> `/Gaze/pos:i`

`/Object/pos:o` -> `/Arm/pos:i`

Inhibition:

`Rest Arm`

`Be Curious`

Modeling Catch and Return scenario